Représentation compacte à base de quadrilatères pour la vidéo 3D

Thomas Colleu, Luce Morin, Claude Labit, Stéphane Pateux, Raphaèle Balter

Représentation compacte à base de quadrilatères pour la vidéo 3D

Compact quad-based representation for 3D video

T. Colleu[1,3] L. Morin[2] C. Labit[1] S. Pateux[3] R. Balter[3]

[1] INRIA Rennes Bretagne Atlantique - Campus de Beaulieu - 35042 Rennes
{thomas.colleu, claude.labit}@irisa.fr
[2] INSA/IETR - 20, Avenue des Buttes de Coësmes - 35043 Rennes
luce.morin@insa-rennes.fr
[3] Orange Labs - 4, rue du Clos Courtel - 35512 Cesson-Sévigné
{thomas.colleu, stephane.pateux, raphaele.balter}@orange-ftgroup.com

Résumé

Cet article se situe dans le contexte de la vidéo 3D. A partir des données d'entrée de type vidéos multi-vues plus profondeur, la représentation proposée à base de polygones 3D prend en compte de façon unifiée les problèmes suivants : compacité, compression et génération de vues intermédiaires. La construction de cette représentation se décompose en deux parties. Tout d'abord, un ensemble de polygones 3D est obtenu en faisant une décomposition en quadtree des cartes de profondeur. Ces polygones permettent de gérer les problèmes identifiés dans l'état de l'art. Ensuite, une élimination sélective des polygones est effectuée afin de réduire les redondances inter-vues et obtenir ainsi une représentation compacte. Des tests sur deux séquences réelles montrent une bonne qualité de rendu pour une quantité d'information qui augmente faiblement comparé à une vidéo mono-vue.

Mots Clés

Vidéo 3D, représentation de données, vidéo multi-vues plus profondeur, quadtree, compression, 3DTV, FTV.

Abstract

The context of this study is 3D video. Starting from multiview video plus depth as input data, the goal of the proposed representation based on 3D polygons is to take into account in a unified manner different issues such as compactness, compression, and intermediate view synthesis. The method used to build such a representation is divided into two parts. First, a set of 3D polygons is obtained thanks to a quadtree decomposition of the depth maps. Polygons allow to deal with the problems identified in the state-of-the-art section. In the second part, a selective elimination of the polygons is performed in order to reduce inter-view redundancies and thus provide a compact representation. Experiments on two real sequences show good quality results at the rendering stage for a small data overload compared to mono-view video.

Keywords

3D video, data representation, multiview video plus depth, quadtree, compression, 3DTV, FTV.

1 Introduction

La vidéo 3D est annoncée comme l'évolution logique de la vidéo 2D. Deux types d'applications de la vidéo 3D peuvent être distinguées. La première, appelée 3DTV (pour *3 Dimensionnal TeleVision*), apporte une sensation de relief aux utilisateurs grâce à un système d'affichage reproduisant la vision binoculaire humaine. De tels systèmes d'affichage existent actuellement et nécessitent donc d'afficher deux vues (écrans stéréoscopiques) ou N=8,10,12... vues (écrans multiscopiques ou multi-vues), certains fonctionnant sans lunettes spécifiques (écrans auto-stéréoscopiques, ou auto-multiscopiques). La deuxième application, appelée FTV (pour *Free-viewpoint TeleVision*), permet à un utilisateur de naviguer de façon interactive dans la vidéo 3D, en choisissant le point de vue désiré. Le degré de navigation possible dépend du type d'application et des données disponibles.

Pour réaliser les deux applications 3DTV et FTV à partir de vidéos réelles, une première approche est d'acquérir l'ensemble des vues nécessaires pour la restitution [19]. Cette approche est celle généralement utilisée pour la vidéo stéréoscopique (2 vues), mais elle se généralise difficilement à N vues en raison de la difficulté d'acquisition. De plus une telle approche interdit l'inter-opérabilité entre acquisition et rendu et entre écrans différents (taille de l'écran, nombre de vues affichées par l'écran, distance par rapport à l'écran). Pour les applications de type FTV, acquérir l'ensemble des vues nécessaires pour la restitution impose un nombre très important de caméras [18] et limite considéra-

blement la navigation.
L'approche alternative est de générer des vues non acquises (vues intermédiaires) à partir des vues acquises. Pour cela, une représentation intermédiaire des informations doit être construite. Le choix de cette représentation doit prendre en considération les différentes contraintes d'un système de vidéo 3D. Elle doit d'une part pouvoir être construite à partir des données d'acquisition de façon générique. D'autre part, les données doivent être transmises sur un canal à débit limité, ce qui nécessite que la représentation soit compacte et puisse être efficacement compressée. Enfin, le nombre de points de vue affichés à l'utilisateur pouvant varier en fonction du système d'affichage utilisé (de deux à quelques dizaines), il faut donc une représentation flexible et qui permette de synthétiser les vues nécessaires avec une bonne qualité, généralement en temps réel.
Les caractéristiques du système considéré sont les suivantes : le nombre de caméras d'acquisition est limité ; aucun a priori n'est porté sur le contenu de la scène ; le rendu de la scène est photo-réaliste ; la navigation autour des vues acquises est limitée.
Dans cet article, nous proposons une représentation de données multi-vues de type MVD (pour *Multi-View plus Depth*), qui sont constituées d'une vidéo multi-vues et des cartes de profondeur associées : une pour chaque vue et pour chaque instant.
En utilisant les données MVD, des vues intermédiaires peuvent être générées, ce qui permet a priori d'espacer les vues d'acquisition et de réduire leur nombre tout en conservant un rendu d'image de bonne qualité. Néanmoins, ces données constituent un ensemble de données très volumineux et redondant, qui n'est pas nécessairement cohérent spatialement et temporellement. De plus, la méthode de génération de vues intermédiaires reste un problème ouvert. Dans le groupe de normalisation ISO-MPEG, le groupe de travail FTV étudie actuellement la représentation et le codage de données du type MVD, ainsi que leur utilisation pour la génération de vues arbitraires dans le but de définir à terme un standard de compression efficace pour la vidéo 3D.
Dans la section 2, différentes représentations de vidéos multi-vues seront présentées et les problématiques concernant les données MVD seront détaillées. Ensuite, nous présenterons une vue globale de la représentation proposée dans la section 3, et nous détaillerons ces principales étapes dans les sections 4 et 5. Enfin, nous présenterons les résultats obtenus dans la section 6.

2 État de l'art

Cette section présente un état de l'art des représentations pour la vidéo 3D. Des informations plus approfondies sur la vidéo 3D pourront être trouvées dans [14, 12, 6].

Représentations. Il existe de nombreuses représentations pour la vidéo 3D et chacune d'entre elles aboutit à un compromis différent entre qualité visuelle, compacité, et liberté de navigation dans la scène. Elles sont souvent classées selon un axe image-géométrie qui montre les différents types de données utilisés (Figure 1).

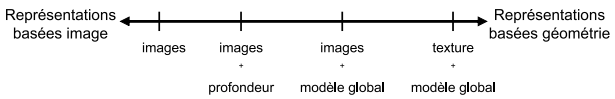


FIGURE 1 – Axe des représentations. Vers la gauche : représentations qui se basent plus sur les images. Vers la droite : représentations qui se basent plus sur la géométrie.

Pour une qualité visuelle élevée, les représentations basées image telles que light field [7] ou ray-space [18] sont les plus appropriées. Dans la méthode light field, le flot de lumière dans la scène est décrit et stocké à partir des images acquises. Pour générer une nouvelle vue, chaque pixel devant être affiché est associé à un rayon de lumière, si ce rayon est dans la base de données alors il est utilisé, sinon, les rayons les plus proches sont sélectionnés et mélangés. Cette représentation nécessite une acquisition très dense des images et donc privilégie la qualité visuelle au détriment de la compacité et de la liberté de navigation.
A l'inverse, les représentations basées sur un modèle géométrique global sont plus appropriées pour une grande liberté de navigation ([2, 5, 8, 17, 13]). En effet, une nouvelle vue est générée par projection perspective des images grâce au modèle géométrique, ce qui permet de réduire la densité d'acquisition des images et donc soit de réduire le nombre d'image, soit de couvrir une plus grande surface de la scène. Néanmoins, le modèle géométrique représente toujours une approximation de la scène plus ou moins forte selon l'échantillonnage choisi et selon les erreurs de reconstruction, ce qui résulte en une dégradation de la qualité des images générées comparé aux représentations basées image.
Pour les applications de vidéo 3D affichées sur écran autostéréoscopique multi-vues, la navigation se limite au déplacement de l'utilisateur par rapport à l'écran. Dans ce contexte, les représentations basées image plus profondeur sont de plus en plus étudiées. Une carte de profondeur est une image qui associe une valeur de profondeur (i.e. la distance à la caméra) à chaque pixel, ce qui en fait une représentation 2D de la surface 3D d'une scène. Les représentations basées image plus profondeur permettent la génération d'images intermédiaires par projection perspective grâce aux cartes de profondeur, ce qui réduit la densité d'acquisition des images. De plus, c'est une représentation flexible car le nombre de vues générées peut-être adapté à de nombreux écrans et ainsi seules les cartes de profondeurs nécessaires sont transmises.
Plusieurs solutions existent à partir de la représentation basée image plus profondeur. La plus simple consiste à n'utiliser qu'un seul couple image plus profondeur par instant temporel (aussi appelé 2D+Z)[4]. Cependant, un seul couple 2D+Z n'est pas équivalent à un couple d'images stéréo car les régions occultées dans une vue ne sont pas

représentées par la carte de profondeur, et par conséquent, lorsqu'une nouvelle vue est générée, ces régions occultées se découvrent. Si la navigation est faible autour de la vue d'origine, alors les petits découvrements peuvent être remplis par filtrage ou par des techniques de remplissage de texture (padding, in-painting).

Pour remplir des découvrements plus importants autour d'une vue, une extension du format 2D+Z a été proposée dans [1]. L'idée est basée sur la représentation LDI (Layered Depth Image) proposée par Shade et al. [15] dans laquelle plusieurs valeurs de couleurs et de profondeurs sont stockées dans des couches qui correspondent aux différentes couches d'occultations d'une vue et qui sont donc ordonnées par ordre de profondeur. Dans [1], une seule couche d'occultation est utilisée. L'avantage est que la qualité visuelle des vues générées ainsi que la liberté de navigation sont améliorées alors que la quantité d'information ajoutée reste faible. Cependant, une seule couche d'occultation peut être insuffisant dans le cas de scènes complexes contenant plusieurs niveaux d'occultations. Récemment, une méthode pour extraire les régions occultées a été présentée dans [11], la représentation obtenue est appelée LDV (Layered Depth Video). Un inconvénient de l'utilisation des couches de profondeur vient du fait qu'elles sont projetées dans la vue de référence et conservées dans cette vue, ce qui cause une perte de résolution dans les zones qui se découvrent et donc réduit la navigation par rapport aux données capturées initialement.

Pour une navigation plus large, la représentation LDV est limitée du fait de l'utilisation d'une seule vue (augmenté par des couches d'occultations) et il est donc nécessaire d'utiliser plusieurs couples 2D+Z. Ces données sont alors appelées MVD (Multi-view Video plus Depth). Avec ces données, une vue intermédiaire est générée à partir des vues gauche et droite ce qui permet de remplir la plupart des régions occultées dans l'une des deux vues et d'obtenir une vue intermédiaire de très bonne qualité ([20, 16]). En revanche, les redondances inter-vues sont généralement élevées car la même scène est capturée depuis plusieurs points de vue, et par conséquent la quantité d'information utilisée n'est pas optimale.

Compression des cartes de profondeur. Les cartes de profondeur sont des images en niveaux de gris. Elles peuvent donc être compressées par un codec vidéo performant du type H.264. Cependant, une carte de profondeur représente la surface d'une scène et possède des propriétés différentes d'une image représentant la texture d'une scène. De ce fait, l'utilisation de cartes de profondeur compressées avec des méthodes adaptées aux images de textures crée des artéfacts visuel gênants, en particulier au niveau des discontinuités de profondeur (contours des objets), comme l'illustre l'étude menée dans [10]. Dans la continuité de cette étude, l'article [9] propose un algorithme de codage de profondeur basé "platelets" qui décompose les cartes de profondeur avec des primitives géométriques respectant les discontinuités ce qui permet d'obtenir une meilleure qualité de rendu pour une efficacité de compression donnée.

Rendu basé profondeur. Le rendu de vue intermédiaire utilisant une carte de profondeur est un rendu basé points : chaque pixel de l'image est d'abord projeté en 3D, ce qui donne un nuage de points 3D, puis projeté dans la vue intermédiaire. Le fait que la géométrie de la scène soit représentée par des points ne permet pas de représenter la nature continue des surfaces dans une scène et cela crée de nombreux pixels non définis qu'il faut ensuite remplir comme dans [16]. Une alternative est de transformer les cartes de profondeur en surface grâce à des primitives géométriques telles que des triangles [20] ou des quadrilatères [3] tout en déconnectant ces primitives dans les zones de discontinuités de profondeurs afin de ne pas coller l'avant-plan et l'arrière-plan de la scène. Cela évite des post-traitements des images générées mais nécessite un processeur graphique. Dans [3], les cartes de profondeurs sont décomposées en quadtree au niveau du rendu afin de minimiser le nombre de polygones tout en conservant la précision géométrique de la scène. Le principe de cette décomposition est de raffiner le quadtree dans les zones de détails géométriques et de représenter les surfaces planes par des plus grand polygones.

3 Représentation proposée

Dans cette étude, les données d'entrée sont de type MVD afin de pouvoir générer des vues intermédiaires. Un exemple de données MVD est illustré dans la figure 2, dans laquelle seules les vues extrême gauche et extrême droite sont montrées sur les 8 vues d'origines.

FIGURE 2 – Exemple de données MVD.

A partir de ce type de données et de l'état de l'art dressé précédemment, plusieurs concepts ont été identifiés concernant la compacité des données [11], la compression des informations de profondeur [9], ainsi que la génération de vues intermédiaires [16, 3]. L'objectif est de proposer une représentation prenant en compte de façon unifiée ces différentes problématiques.

Cette représentation est basée sur un ensemble de polygones 3D :

– Un polygone est associé à un bloc de pixels dans une vue.
– Des informations de profondeur sont associées aux sommets des polygones
– La texture associée à un polygone correspond au bloc de pixels dans la vue considérée.

L'utilisation de primitives géométriques polygonales présente plusieurs avantages. Tout d'abord, ces primitives peuvent s'obtenir par découpage des cartes de profondeur à l'aide d'un algorithme de type quadtree, ce qui permet de limiter le nombre de polygones utilisés afin d'avoir une représentation compacte et un rendu efficace (de façon similaire à [3]). Par ailleurs, l'utilisation de primitives géométriques polygonales permet de mettre en place un système de compression avec pertes des informations de profondeur offrant un meilleur rendu visuel comparé à une compression de type H.264 pour une qualité de compression donnée, comme discuté dans [9]. Ensuite, l'utilisation de polygones est une méthode usuelle de rendu (e.g. [20, 3]) qui permet de proposer une technique de projection des données sans avoir à recourir à des post-traitements (e.g. [16]) pour remplir les pixels non définis dûs aux projections basées points. La génération de vues intermédiaires peut ainsi être implémentée de façon très efficace via des librairies graphiques telles que OpenGL. Contrairement à [20] et [3] où les primitives sont créées seulement à l'étape de rendu à partir des cartes de profondeur, la représentation proposée utilise directement ces primitives géométriques, avant même l'étape de codage. L'extraction des polygones à partir des cartes de profondeur sera présentée dans la section 4. Enfin, une technique d'élimination sélective des polygones permet de réduire les redondances entre les vues (de façon similaire à [11]), et également de limiter les défauts de texture au niveau des frontières de discontinuité (aussi appelés artefacts de ghosting) dus à un mélange des textures de l'avant-plan et de l'arrière-plan. Il est à noter que les textures des polygones sont toutes conservées dans leur vue d'origine afin de préserver leurs résolution, contrairement aux représentations du type LDV utilisant des couches d'occultation projetées dans une vue de référence. La technique d'élimination des polygones sera présentée dans la section 5.

La figure 3 synthétise le principe de construction de la représentation proposée ainsi que son utilisation dans un cadre applicatif (compression et rendu d'un nombre de vues différent).

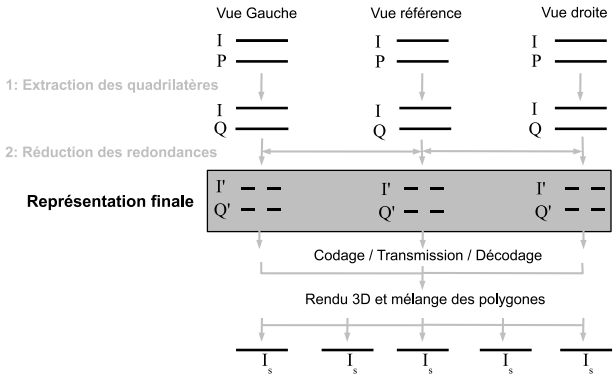


FIGURE 3 – Construction de la représentation. I : image, P : profondeur, Q : quadrilatères, I'/Q' : image/quadrilatères réduits, I_s : image synthétisée

4 Extraction des quadrilatères

Dans cette partie, chaque carte de profondeur est traitée indépendamment afin d'obtenir un ensemble de polygones 3D par vue. Le type de polygone qui a été choisi est le quadrilatère. Ces quadrilatères sont définis par des blocs de pixels dans chaque vue. Ce choix est motivé par différents aspects. D'une part, l'utilisation d'une technique de type quadtree permet de réaliser un découpage en blocs de taille adaptative permettant de gérer le compromis entre la fidélité des données de profondeur et la compacité de ces données. De plus, dans un contexte de codage de vidéo 3D, le codage des informations de texture sera très probablement basé sur une technique de codage proche de celle utilisée pour la vidéo classique, à savoir une technique de codage par bloc. On pourra alors ainsi profiter de la cohérence du codage par bloc des informations de profondeur et de texture.

L'obtention des quadrilatères pour une carte de profondeur se fait à l'aide d'une technique de type quadtree. Un exemple de découpage en blocs d'une carte de profondeur obtenu avec une telle technique est montré dans la figure 4. La technique de découpage en quadtree utilisée ici s'effectue en deux temps :

Gestion des discontinuités. Dans un premier temps, les blocs de l'image sont découpés tant que l'on observe deux pixels voisins dans le bloc considéré ayant une différence de profondeur supérieure à un certain seuil. Soit deux pixels voisins $p1$ et $p2$ dans un bloc B et leurs profondeurs respectives Z_{p1} et Z_{p2} et un seuil de discontinuité S_d, le bloc sera découpé si :

$$\max_{p1,p2 \text{ voisins } \in B}(abs(Z_{p1} - Z_{p2})) > S_d$$

A ce stade, on obtient une première version grossière de la représentation.

Fidélité aux données de profondeur. Dans un second temps, les blocs de la représentation grossière sont découpés de façon itérative tant que le quadrilatère tridimensionnel obtenu, qui n'est pas forcément plan, ne représente pas avec une qualité suffisante les données de profondeur d'entrée. Pour estimer cette qualité, le plan π_B qui approxime aux moindres carrés les valeurs de profondeur du bloc B est estimé. Ensuite, la distance maximum entre les valeurs de profondeur du bloc et le plan est calculée. Pour chaque pixel p d'un bloc B et un seuil d'erreur au

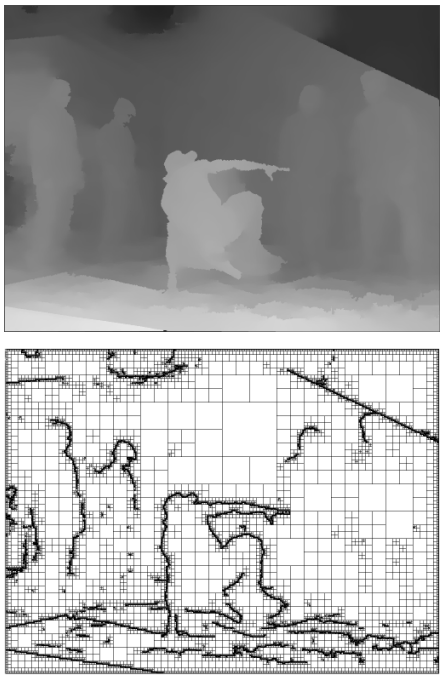


FIGURE 4 – Illustration du découpage en blocs par une technique de type quadtree. Haut : Carte de profondeur. Bas : Découpage en blocs

plan S_p, le bloc sera découpé si :

$$\underset{p \in B}{max}(dist(p, \pi_B)) > S_p$$

où $dist(p, \pi_B)$ est la distance entre le pixel de profondeur Z_p et le plan π_B. Lorsqu'un bloc satisfait le critère ci-dessus, alors un quadrilatère Q_B y est associé à l'aide des valeurs de profondeur aux sommets du bloc. Bien que ce quadrilatère ne soit pas obligatoirement plan, la distance maximum entre Q_B et π_B sera inférieure ou égale au seuil S_p ce qui permet de contrôler la qualité de représentation du quadrilatère par rapport aux données. La figure 5 illustre l'approximation par un plan d'un bloc de pixels projetés en 3D ainsi que le quadrilatère obtenu pour ce bloc.

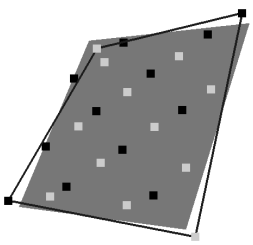


FIGURE 5 – Approximation par un plan d'un bloc de pixels projeté en 3D et quadrilatère obtenu pour ce bloc. Les points noirs sont devant le plan et les points gris derrière.

5 Réduction des redondances

Dans cette partie, les redondances entre les différentes vues de la vidéo pour un instant temporel donné sont considérées. En effet, de larges régions capturées de la scène sont généralement communes à plusieurs vues, ce qui introduit de nombreuses redondances inter-vues qui peuvent être réduites. Dans ce but, une élimination sélective des quadrilatères obtenus précédemment est effectuée. Cette opération permet également de limiter les défauts de texture au niveau des frontières de discontinuité (artefacts de ghosting). En sortie de cet algorithme, l'élimination des quadrilatères se traduit par un étiquetage des branches de chaque quadtree selon qu'elles sont conservées ou éliminées. Ainsi, lors de l'étape de codage des données, les branches éliminées des quadtree et les blocs de texture correspondants ne seront pas codés.

Pour expliquer les opérations effectuées, les notations choisies sont : N le nombre de vues ; V_n la vue numéro n, $n \in \{1..N\}$; Q_n l'ensemble des quadrilatères de V_n ; et E l'ensemble des quadrilatères conservés dans toutes les vues.

L'idée principale est d'initialiser E avec les quadrilatères d'une vue de référence V_r qui est généralement la vue centrale, puis de compléter et modifier E avec les quadrilatères des autres vues de façon itérative. A chaque itération, une nouvelle vue V_n est sélectionnée et deux opérations sont effectuées : la première est la réduction dans Q_n des redondances avec E. On obtient alors Q'_n qui est ajouté à E. La deuxième est la réduction dans E des superpositions introduites par l'opération précédente. La méthode générale est résumée dans l'algorithme 1.

Algorithme 1 Réduction des redondances

ENTRÉES: N ; $Q_n, n \in \{1..N\}$; $r, 1 \leq r \leq N$
SORTIE: E

1: $E \leftarrow Q_r$
2: **pour** $n = 1$ à $N, n \neq r$ **faire**
3: $Q'_n \leftarrow$ Réduire_Redondances(Q_n,E)
4: $E \leftarrow E \cup Q'_n$
5: $E \leftarrow$ Réduire_Superpositions(E)
6: **fin pour**

Les deux opérations spécifiques (étapes 3 et 5 de l'algorithme) sont détaillées dans les sections suivantes. Ces deux opérations utilisent la technique de projection perspective afin d'identifier les régions redondantes entre les différentes vues. Le principe est de transformer tout d'abord les quadrilatères 2D d'une vue V_x en quadrilatères 3D en utilisant la profondeur des sommets et la projection perspective inverse, puis de projeter ces quadrilatères 3D dans l'une des autres vues V_y. Il est ainsi possible de comparer les informations de V_x projetées dans V_y avec les informations originales de V_y.

Réduction des redondances dans Q_n. L'étape 3 de l'algorithme est à présent expliquée. L'objectif est de réduire

dans Q_n les redondances avec E par élimination de quadrilatères. Pour cela, les quadrilatères contenus dans E dont la taille est supérieure à un seuil S_t sont tout d'abord projetés dans la vue V_n, ce qui donne une image contenant des zones de découvrement autour des discontinuités et des limites de E, comme illustré dans la figure 6(haut). Ensuite, ces zones de découvrement permettent d'identifier quels quadrilatères de Q_n peuvent être supprimés. Le test est le suivant : pour chaque quadrilatère de Q_n, si le bloc de pixels qu'il forme dans l'image n'est pas situé sur une zone de découvrement, alors ce quadrilatère est éliminé car il est redondant avec les données de E. L'ensemble des quadrilatères conservés est noté Q'_n et illustré dans la figure 6(bas). Cette étape permet donc de réduire les redondances tout en remplissant les zones de découvrement. Le fait de n'avoir projeté que les quadrilatères de E ayant une taille supérieure à S_t a pour effet de créer des zones de découvrements plus larges. Par conséquent, une superposition entre les quadrilatères de Q'_n et ceux de E a été créée. Le but de cette superposition est de pouvoir éliminer les petits quadrilatères de E qui sont généralement peu fiables, cette étape est décrite dans la section suivante.

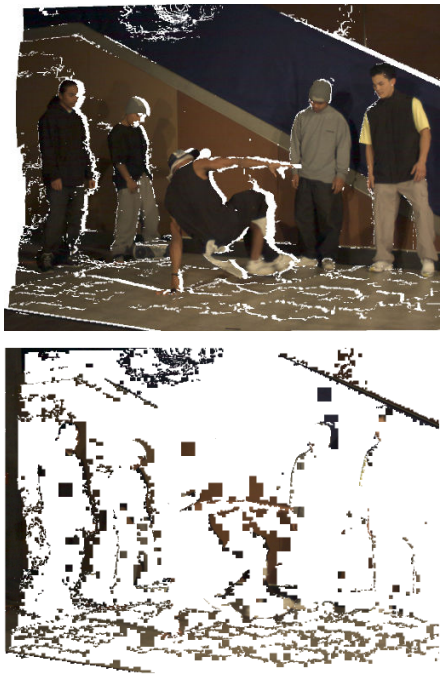


FIGURE 6 – Réduction des redondances dans Q_n. Haut : Découvrements obtenus par projection de E dans V_n. Bas : Quadrilatères conservés (Q'_n)

Réduction des superpositions dans E. L'étape 5 de l'algorithme est à présent expliquée. Elle consiste à réduire dans E les superpositions dues aux quadrilatères de Q'_n qui viennent d'être ajoutés. L'idée est de tester si les quadrilatères de Q'_n recouvrent entièrement certains quadrilatères de E. Si c'est le cas, alors le quadrilatère concerné peut être éliminé. Pour tester si un quadrilatère q_m appartenant à E et venant de la vue V_m est recouvert, Q'_n est d'abord projeté dans V_m. Ensuite, si le bloc de pixel formé par le quadrilatère q_m est entièrement recouvert par des quadrilatères de Q'_n (un test de profondeur est effectué ici), alors ce quadrilatère est éliminé. La figure 7 illustre cette technique. L'image 7(a) montre la projection de E dans une vue avant d'effectuer l'étape 5. Un artefact de ghosting apparaît : le contour d'un objet en avant-plan apparaît au milieu du mur dans l'arrière-plan. L'image 7(b) montre les quadrilatères superposés. Les quadrilatères noirs appartiennent à Q'_n et les quadrilatères gris sont les autres quadrilatères de E. La région du ghosting correspond bien à des petits quadrilatères le long d'une discontinuité. Ces quadrilatères sont superposés avec d'autres venant de Q'_n. Les images (c) et (d) montre les résultats de l'étape 5. Les superpositions ont été réduites ce qui a permis de réduire le ghosting et les redondances de données.

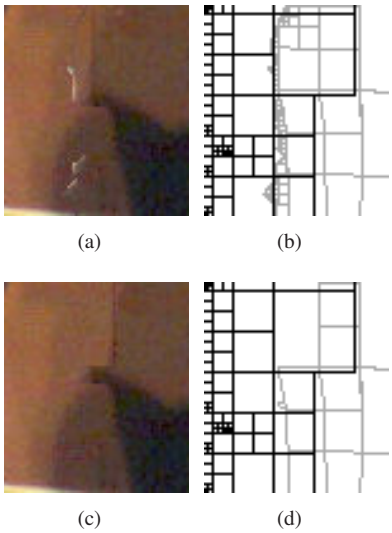
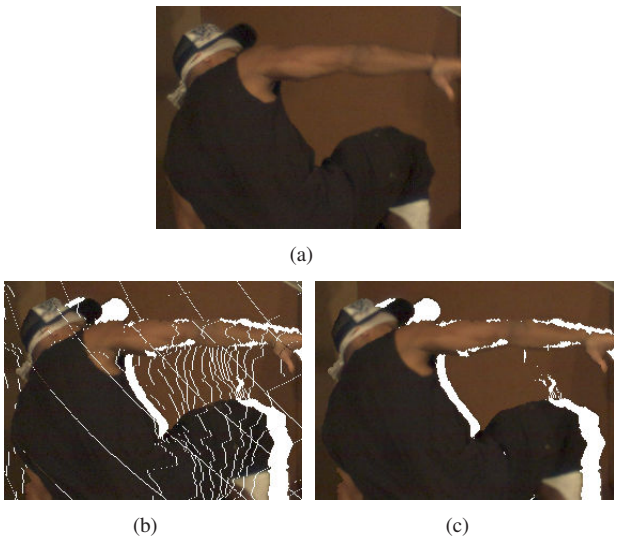


FIGURE 7 – Réduction des superpositions et de l'effet de ghosting. Les quadrilatères noirs appartiennent à Q'_n et les gris sont les autres quadrilatères de E.

6 Résultats

Les tests ont été effectués avec les séquences MVD *Breakdancer* et *Ballet* mises à disposition librement par Microsoft. Ces séquences ont été acquises avec un système comportant 8 cameras de résolution 1024x768 disposées sur un arc horizontal. Le champ de vision est d'environ 30°. Les cartes de profondeurs ont été estimées avec un algorithme de reconstruction stéréo basé sur une segmentation des couleurs et permettant d'obtenir des correspondances photo-consistantes dans toutes les vues [20]. Les seuils utilisés pour ces tests ont été choisis de manière empirique.
L'extraction des quadrilatères a été effectuée avec les seuils suivant : seuil de discontinuité $S_d = 3$ pour des valeurs de profondeurs codées sur 8 bits ; seuil de fidélité aux données de profondeur $S_p = 0.6$. La figure 4 donnée en section 4 illustre la décomposition en quadtree obtenue. Le

nombre moyen de quadrilatères pour les séquences *Breakdancer* et *Ballet* est de 47000 par vue. Ce chiffre peut être comparé au nombre de pixels de la carte de profondeur qui est de 786432. Il a y donc un rapport de 17 entre le nombre de primitives géométriques utilisées pour une carte de profondeur et pour la représentation à base de quadrilatères. A partir de ces quadrilatères et de la texture correspondante, des vues intermédiaires peuvent être synthétisées. Pour comparer la qualité des images obtenues entre un rendu basé points (à partir d'une carte de profondeur) et un rendu basé quadrilatères, la figure 8 donne un extrait d'une zone zoomée sur le danseur central de la séquence *Breakdancer*. L'image (a) montre les données originales de la vue V_1, et les images (b) et (c) montrent les résultats de la projection de V_2 dans V_1 à base de points (b), puis à base de quadrilatères (c). Les zones de découvrement apparaissent en blanc. Dans l'image (c), la continuité de la surface a été respectée et seuls les grandes discontinuités de profondeur font apparaitre des découvrements. Au contraire, dans l'image (b), de nombreuses discontinuités apparaissent même quand la surface de la scène est continue. La représentation à base de quadrilatères permet donc d'éviter une étape de post-traitement qui serait nécessaire avec la représentation basée points. Dans les deux images (b) et (c), aucune dégradation n'est visible par rapport à l'image (a) en dehors des découvrements.



(a)

(b) (c)

FIGURE 8 – Comparaison entre l'image originale (a), le rendu basé points (b) et le rendu basé quadrilatères (c)

La réduction des redondances inter-vues a été effectuée avec le seuil sur la taille des quadrilatères $S_t = 2$, c'est-à-dire un bloc dans l'image de taille 2x2. Les résultats suivants sont donnés pour une configuration à trois vues (V_1, V_2, V_3) dans laquelle la vue centrale V_2 est choisie comme référence. Le tableau 1 compare le nombre de quadrilatères utilisés en fonction des différentes étapes de la réduction des redondances. La première ligne du tableau correspond aux vues complètes telles qu'elles sont obtenues

	BreakDancer	*Ballet*
$V_1 + V_2 + V_3$	129960	152259
étape 3 seule	74277	90118
étape 3 + étape 5	54436	62902

TABLE 1 – Comparaison du nombre de quadrilatères en fonction des étapes de la réduction des redondances

après l'extraction des quadrilatères ; la deuxième ligne correspond à l'étape 3 de l'algorithme, c'est-à-dire la réduction des redondances dans les vues V_1 et V_3 ; La troisième ligne correspond à l'algorithme complet (étape 3 + étape 5). Il y a donc un rapport entre le nombre de quadrilatères avant et après l'algorithme égal à 2,4. De plus, pour une interprétation plus parlante, en comparant le nombre de quadrilatères moyen pour une seule vue avant la réduction (47000) et celui pour les 3 vues après la réduction (59000), l'augmentation de quadrilatères représente 26%, soit 13% par vue supplémentaire. Finalement, à partir de la représentation obtenue après la réduction des redondances, il est possible de générer les vues V_1, V_2 ou V_3 avec une bonne qualité. La figure 9 montre la génération de la vue V_1. Pendant cette génération de vue, lorsque plusieurs quadrilatères sont projetés dans une même partie de l'image, alors leurs contributions sont mélangées de façon égale ce qui permet d'adoucir les incohérences de texture entre quadrilatères venant de vues différentes. En résultat, une image de bonne qualité est obtenue. Cependant, des dégradations peuvent apparaître comme des différences de couleurs (e.g. dans la zone d'ombre près de la seconde personne en partant de la droite) ou comme les contours du danseur qui sont marqués de façon exagérées. De plus, la qualité des vues synthétisées dépend des cartes de profondeur utilisées en entrée et qui peuvent contenir des erreurs ou des inconsistances à travers les différentes vues. Pour générer des vues intermédiaires (entre V_1 et V_2 ou bien entre V_2 et V_3) un traitement supplémentaire de type remplissage de texture serait nécessaire afin de remplir les zones de découvrements qui ne sont visibles dans aucune des vues.

FIGURE 9 – Représentation finale rendue dans V_1.

7 Conclusion

Cet article présente une nouvelle représentation compacte à base de quadrilatères pour la video 3D. Cette représentation prend en compte de façon unifiée différents problèmes identifiés dans la littérature : compacité des données, compression des cartes de profondeur, génération de vues intermédiaires. Un ensemble de quadrilatères est extrait grâce à une décomposition en quadtree des cartes de profondeur, et les redondances inter-vues sont réduites par une élimination sélective de quadrilatères. Les résultats ont montré que l'utilisation de polygones apporte un bon compromis entre qualité de rendu des vues générées et compacité des données. De plus, l'étape de réduction des redondances a montré dans le cadre de ces premières expérimentations que pour une configuration à 3 vues, l'augmentation des données par rapport à une configuration mono-vue est de 13% par vue supplémentaire.

De nombreux travaux doivent encore être réalisés. Une étude précise des dégradations visuelles en fonction du nombre de polygones et aussi en fonction de l'éloignement de la vue centrale est nécessaire. De plus la construction de la représentation peut être améliorée afin de gérer les erreurs et les incohérences de profondeurs et de textures entre les polygones des différentes vues. Ensuite, une étude du codage de la représentation proposée doit être menée afin de rendre compte des avantages apportés par la compacité de la représentation. Enfin, la prise en compte de la dimension temporelle des séquences vidéo permettra d'accroître les performances.

Références

[1] W.H.A. Bruls, C. Varekamp, R.K. Gunnewiek, B. Barenbrug, and A. Bourge. Enabling introduction of stereoscopic (3d) video : Formats and compression standards. In *ICIP*, pages 89–92, 2007.

[2] J. Carranza, C. Theobalt, M. A. Magnor, and H. Seidel. Free-viewpoint video of human actors. *ACM Trans. Graph.*, 22(3) :569–577, 2003.

[3] J. Evers-Senne, J. Woetzel, and R. Koch. Modelling and rendering of complex scenes with a multicamera rig. In *Conference on Visual Media Production (CVMP)*, 2004.

[4] C. Fehn, P. Kauff, M. Op de Beeck, F. Ernst, W. IJsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, and I. Sexton. An evolutionary and optimised approach on 3d-tv. In *In Proceedings of International Broadcast Conference*, pages 357–365, Amsterdam, Netherlands, 2002.

[5] J.S. Franco and E. Boyer. Exact polyhedral visual hulls. In *in British Machine Vision Conference (BMVC'03)*, pages 329–338, 2003.

[6] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang. Multiview imaging and 3dtv. *IEEE Signal Processing Magazine*, 24(6) :10–21, November 2007.

[7] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH '96*, pages 31–42, New York, NY, USA, 1996. ACM Press.

[8] T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara. Real-time 3d shape reconstruction, dynamic 3d mesh deformation, and high fidelity visualization for 3d video. *Comput. Vis. Image Underst.*, 96(3) :393–434, 2004.

[9] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Muller, P.H.N. de With, and T. Wiegand. The effect of depth compression on multiview rendering quality. In *3DTV Conference*, 2008.

[10] P. Merkle, A. Smolic, K. Muller, and T. Wiegand. Multi-view video plus depth representation and coding. *ICIP*, 1 :201–204, 2007.

[11] K. Müller, A. Smolic, K. Dix, P. Kauff, and T. Wiegand. Reliability-based generation and view synthesis in layered depth video. In *MMSP*, pages 34–39, 2008.

[12] H.M. Ozaktas and L.Onural. *Three-Dimensional Television. Capture, Transmission and Display*. Springer, 2008.

[13] D. Ruiz. *A distributed and scalable architecture for real time volumetric reconstruction of arbitrary shapes exploiting inter-frame redundancy*. PhD thesis, Université Catholique de Louvain, 2008.

[14] O. Schreer, P. Kauff, and T. Sikora. *3D Videocommunication : Algorithms, concepts and real-time systems in human centred communication*. Book, John Wiley & Sons, 2005.

[15] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *ACM SIGGRAPH*, pages 231–242, 1998.

[16] A. Smolic, K. Muller, K. Dix, P. Merkle, P. Kauff, and T. Wiegand. Intermediate view interpolation based on multiview video plus depth for advanced 3d video systems. In *ICIP*, pages 2448–2451, 2008.

[17] J. Starck and A. Hilton. Surface capture for performance-based animation. *IEEE Comput. Graph. Appl.*, 27(3) :21–31, 2007.

[18] M. Tanimoto. Overview of free viewpoint television. *Signal Processing : Image Communication*, Volume 21, Issue 6 :454–461, 2006.

[19] M. Wojciech and H. Pfister. 3d tv : a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *ACM Trans. Graph.*, 23 :814–824, 2004.

[20] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3) :600–608, 2004.